Lecture 5 – Adders

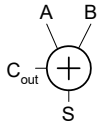
11.2

Adapted from David Harris, HMC, Blaauw, Zhang, UM and others

1

Single-Bit Addition

Half Adder



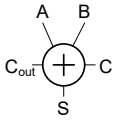
A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

AND XOR

Full Adder

$S = A \oplus B \oplus C$

$C_{out} = MAJ(A, B, C)$



	A	B	C	C_{out}	S
K	0	0	0	0	0
	0	0	1	0	1
P	0	1	0	0	1
	0	1	1	1	0
	1	0	0	0	1
	1	0	1	1	0
G	1	1	0	1	0
	1	1	1	1	1

2

PGK

- For a full adder, define what happens to carries (in terms of A and B)
  - Generate: $C_{out} = 1$ independent of C_{in}
    - G = A • B
  - Propagate: $C_{out} = C_{in}$
    - P = A ⊕ B
  - Kill: $C_{out} = 0$ independent of C_{in}
    - K = ~A • ~B

3

Express Sum and Carry as a function of P, G, K

Generate (G) = AB

Propagate (P) = A⊕ B

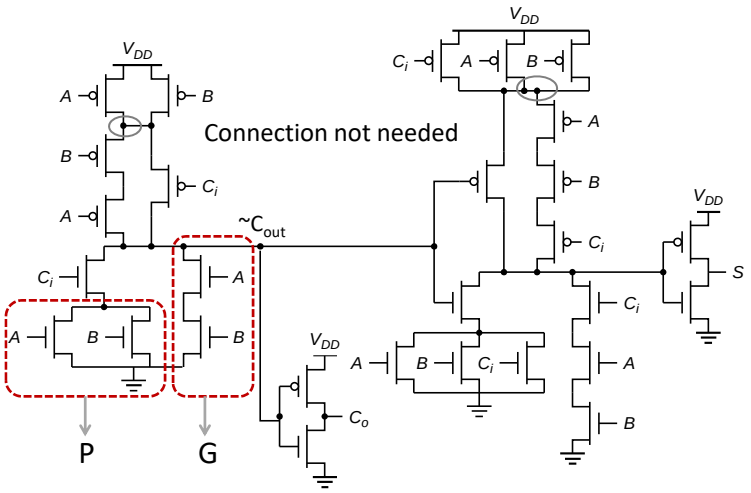
$$C_o(G,P) = G + PC_i$$
$$S(G,P) = P \oplus C_i$$

Note that we will be sometimes using an alternate definition for
Propagate (P) = A+ B

Which is preferred ? A + B OR A ⊕ B

4

Basic Static CMOS Full Adder

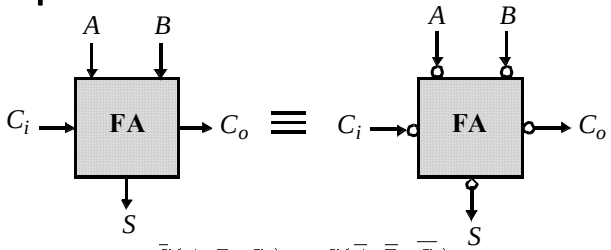


28 Transistors

$S = ABC_i + (A + B + C_i)(\sim C_{out})$

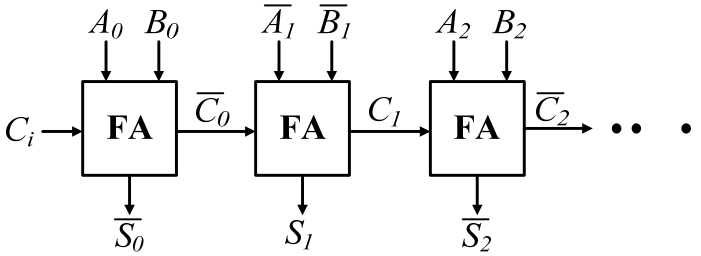
5

First Optimization: Inversion Property



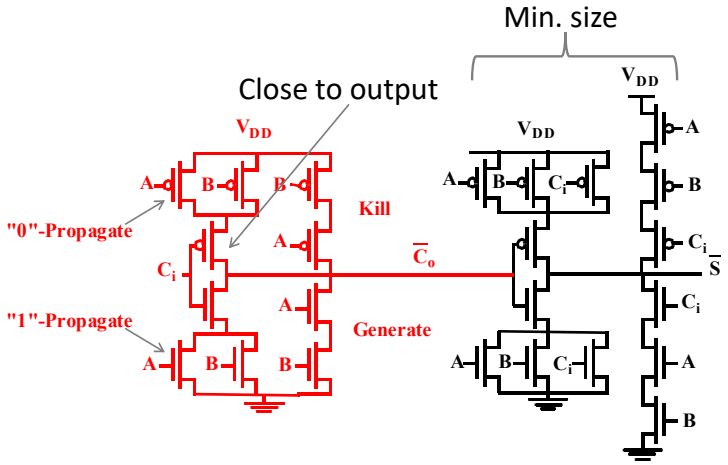
$$\bar{S}(A,B,C_i) = S(\bar{A},\bar{B},\bar{C}_i)$$

$$\overline{C_o}(A,B,C_i) = C_o(\bar{A},\bar{B},\bar{C}_i)$$



6

Second Optimization: The Mirror Adder



24 transistors

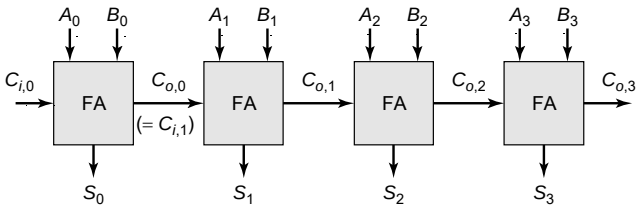
7

The Mirror Adder

- **NMOS and PMOS chains are completely symmetric. Max of two series transistors in carry-generation circuitry.**
- **When laying out cell, most critical issue is minimizing capacitance at node C_o. Reduction of diffusion capacitances is particularly important.**
- **Capacitance at node C_o is composed of 4 diffusion capacitances, 2 internal gate capacitances, and 6 gate capacitances in connecting adder cell**
- **Transistors connected to C_i are placed closest to output**
- **Only transistors in carry stage have to be optimized for optimal speed. All transistors in sum stage can be min size.**

8

The Ripple-Carry Adder



Worst case delay linear with the number of bits

$t_p = O(N)$

$t_{adder} = (N-1)t_{carry} + t_{sum}$

Goal: Make the fastest possible carry path circuit

9

Carry-Ripple using P and G

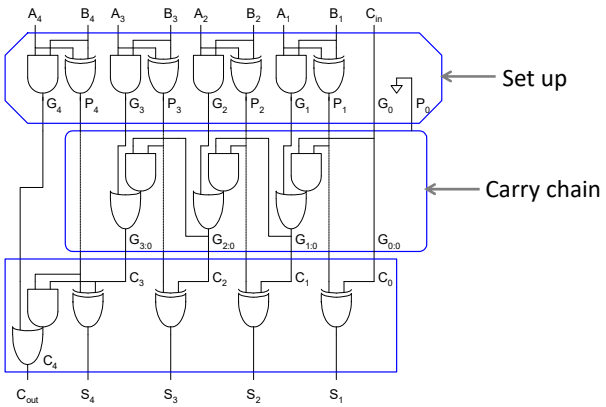
$G_{i:0}$

$C_{i:0} = G_i + P_i \cdot C_{i-1:0}$ $G_{i-1:0}$

$G_{0:0} = C_{in}$

$P_{0:0} = 0$

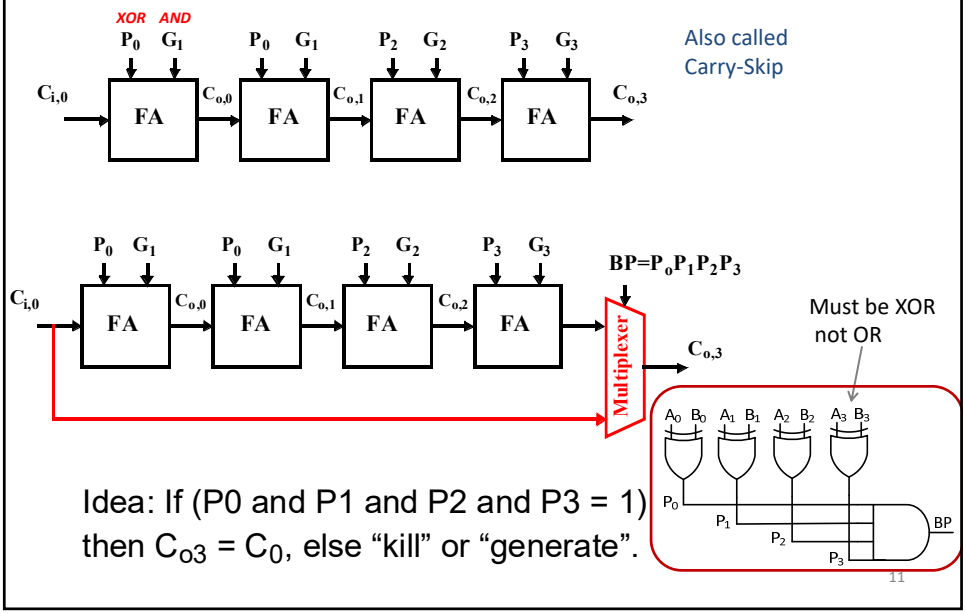
$C_{out,i} = G_{i:0}$



$t_{adder} = t_{setup} + N\text{-}1\ t_{carry} + \max(t_{carry}, t_{sum})$

10

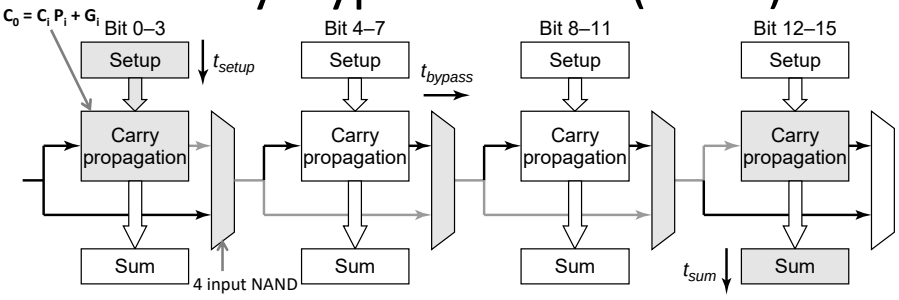
Carry-Bypass Adder

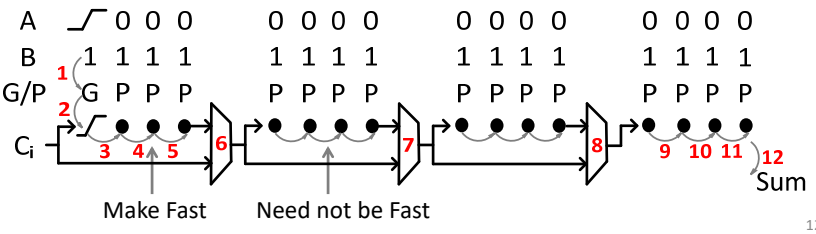


Idea: If (P0 and P1 and P2 and P3 = 1) then $C_{o3} = C_0$, else "kill" or "generate".

11

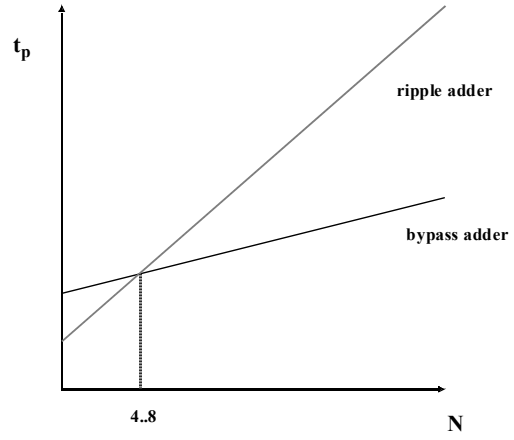
Carry-Bypass Adder (cont.)





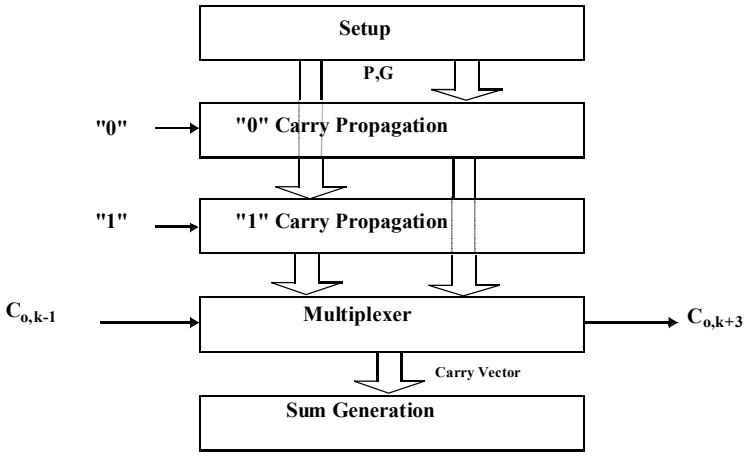
12

Carry Ripple versus Carry Bypass



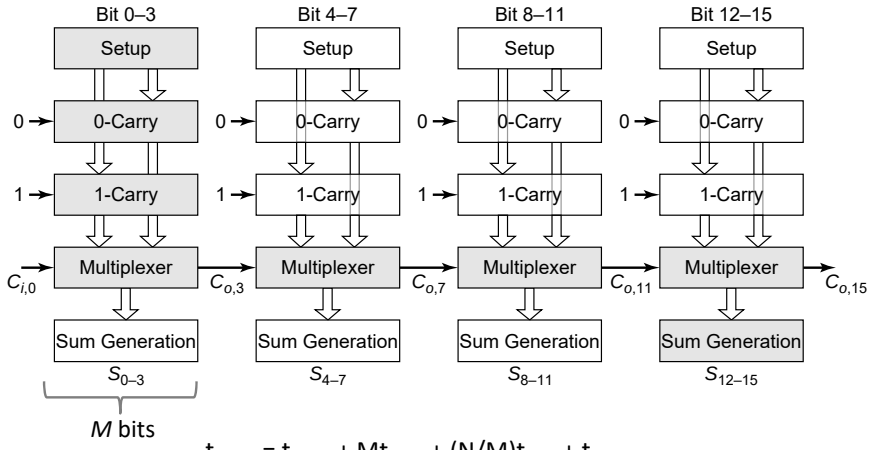
13

Carry-Select Adder



14

Carry Select Adder: Critical Path

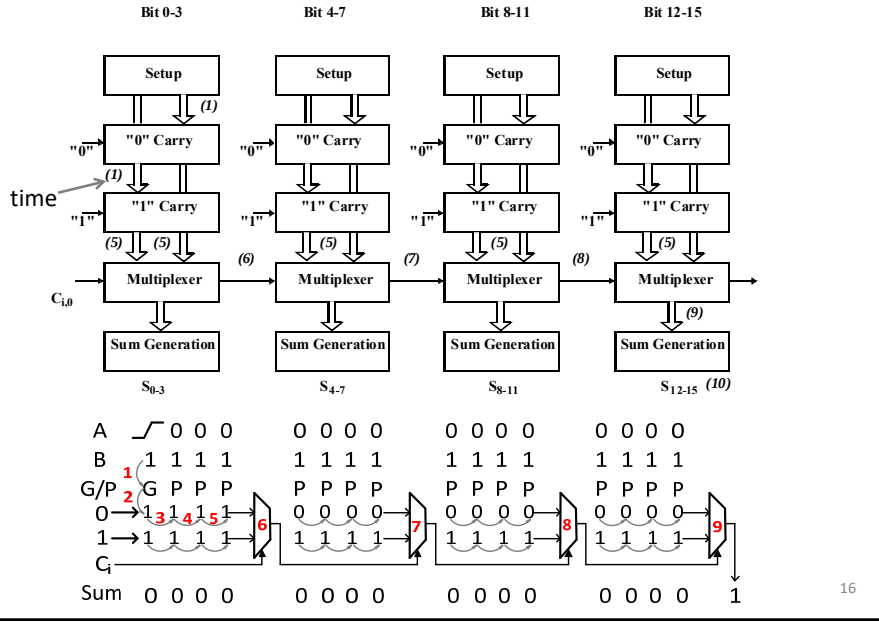


$$t_{adder} = t_{setup} + Mt_{carry} + (N/M)t_{mux} + t_{sum}$$

Can be pre-computed when muxes fire

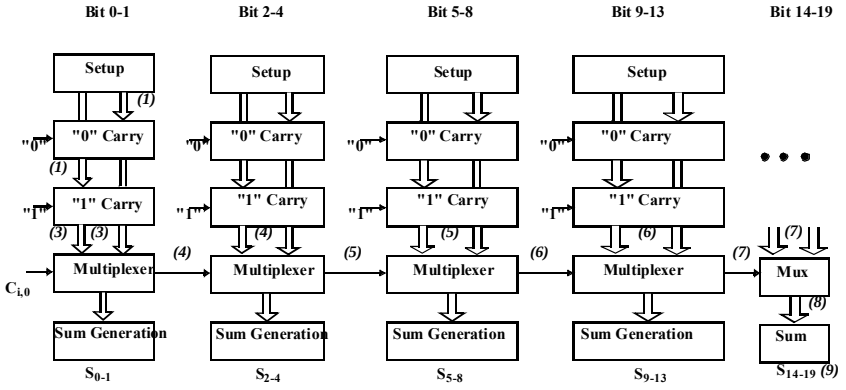
15

Linear Carry Select



16

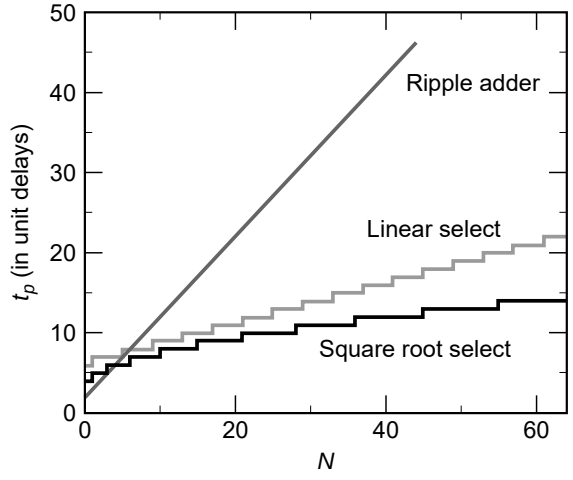
Square Root Carry Select



$$t_{adder} = t_{setup} + Mt_{carry} + sqrt(2N)t_{mux} + t_{sum}$$

17

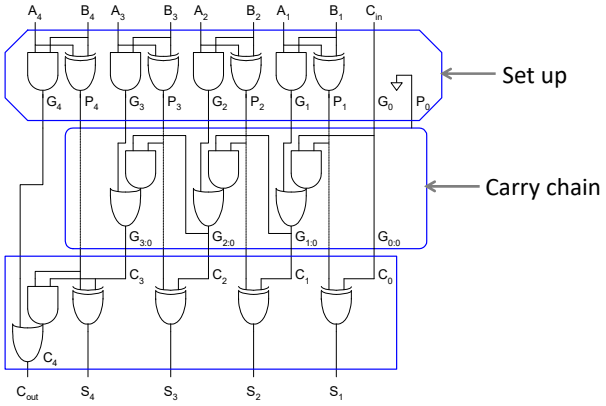
Adder Delays - Comparison



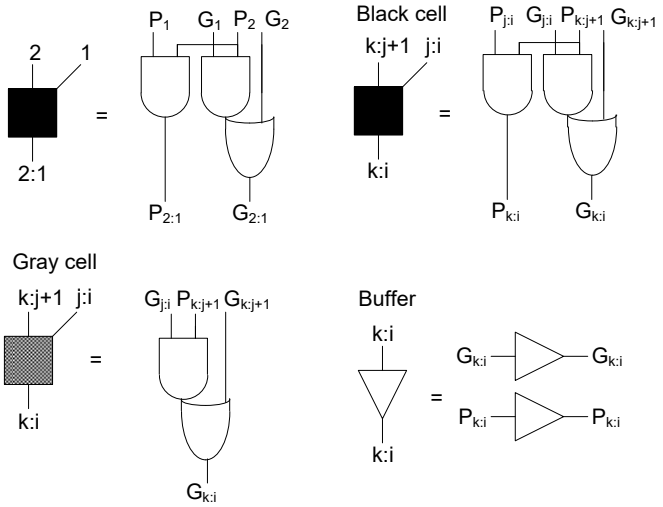
18

Carry-Ripple Revisited

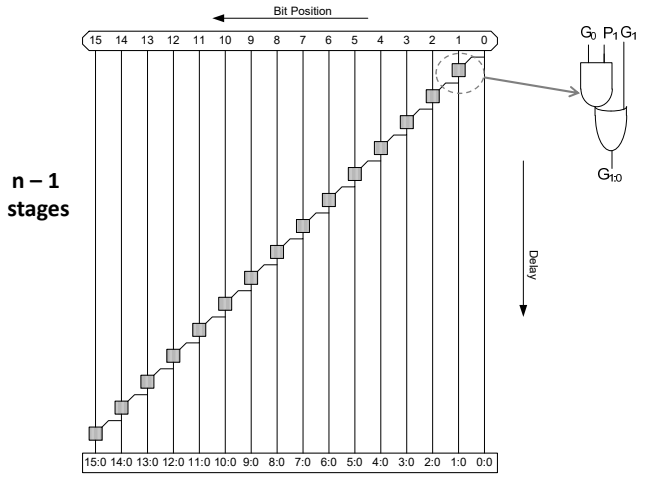
$$C_{i:0} = G_i + P_i \cdot C_{i-1:0}$$
$$G_{0:0} = C_{in}$$
$$P_{0:0} = 0$$
$$C_{out,i} = G_{i:0}$$



PG Diagram Notation

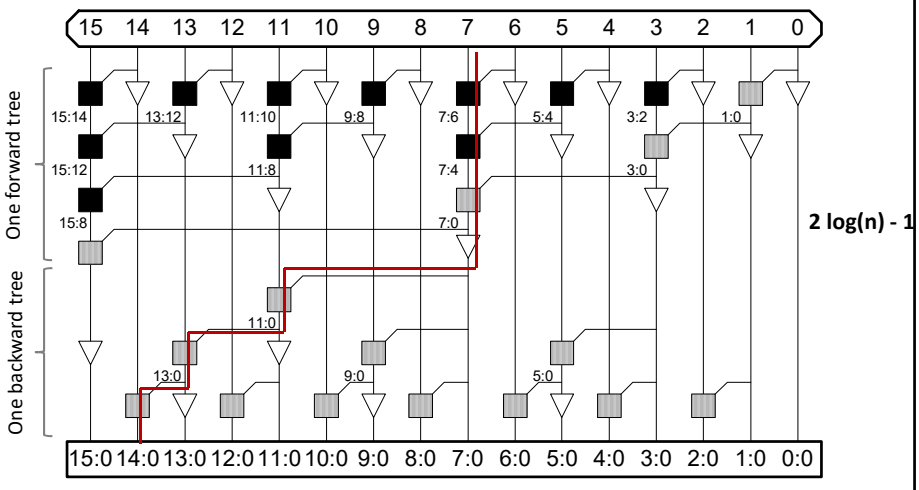


Carry-Ripple PG Diagram



21

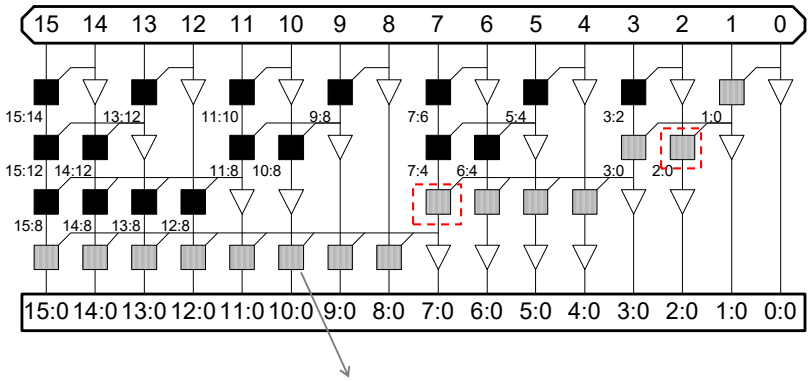
Brent-Kung



22

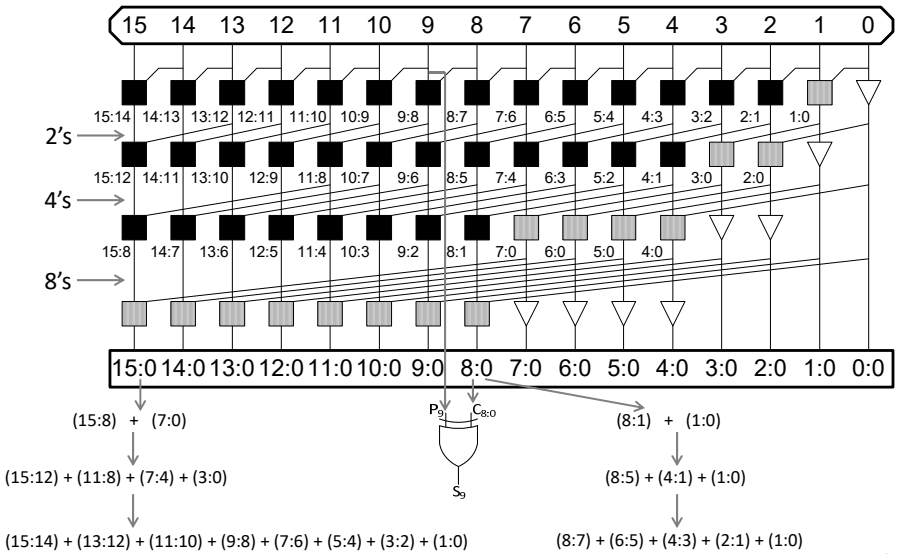
Sklansky

$\log_2(n)$



- Uneven sizing
- Large fanout

Kogge-Stone



Tree Classification

- Logic levels
- Fanout
- Wiring

} Power/Area

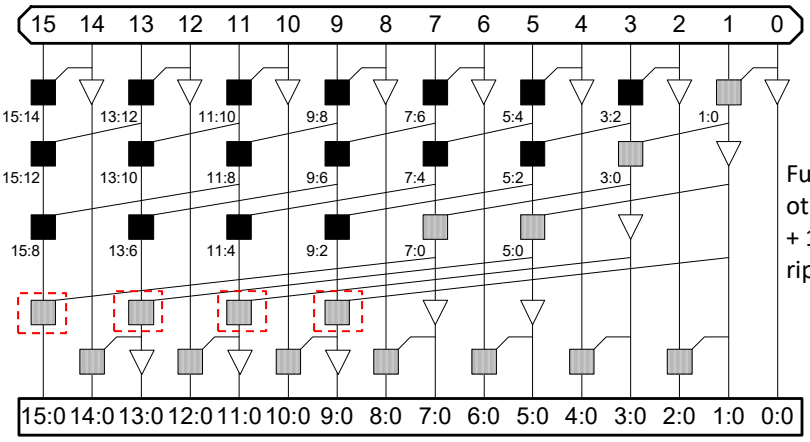
- Kogge-Stone: low logic levels, low fanout, high wiring
- Brent-Kung: low fanout, low wiring, high logic levels
- Sklansky: low logic levels, low wiring, high fanout

25

Han-Carlson

KS + BK

$\log_2(n) + 1$



Full tree every other
+ 1 stage ripple

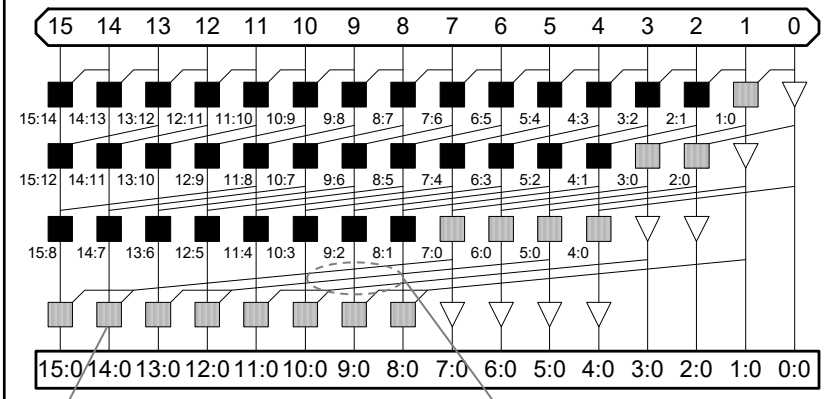
Low fanout, tradeoff between logic levels and wiring

Reduces wire length by half ! → half power compared to Kogge Stone

26

Knowles

KS + Skl



(14:7) + (7:0)

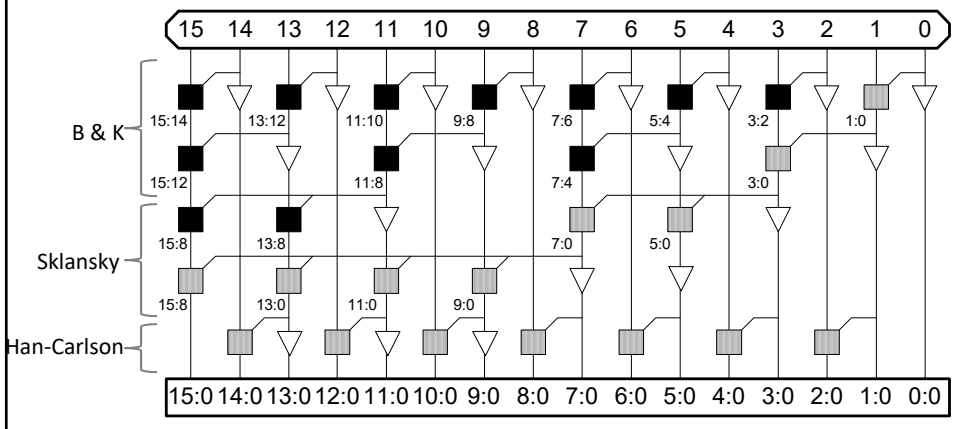
Half the wires by overlapping neighbors →
¼ wire reduction, more fanout

Low logic levels, tradeoff between fanout and wiring

27

Ladner-Fischer

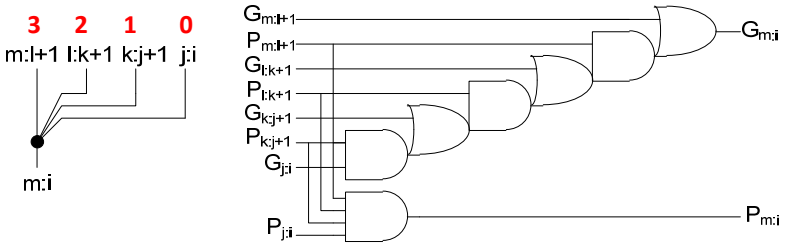
BK + Skl



Low wiring, tradeoff between logic levels and fanout

28

Higher Radix

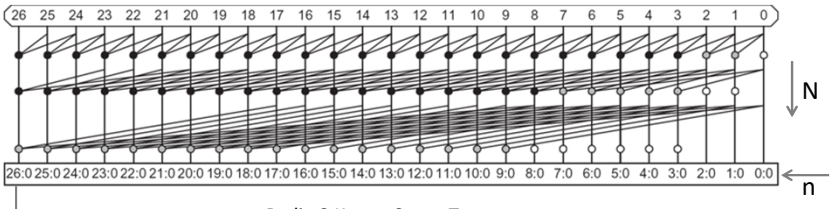


$G = G_3 + P_3 \cdot (G_2 + P_2 (G_1 + P_1 \cdot G_0))$

29

Radix-3 Kogge-Stone

- Fewer logic levels
- **Each stage has more delay**



Radix-3 Kogge-Stone Tree

(26:18) + (17:9) + (8:0)

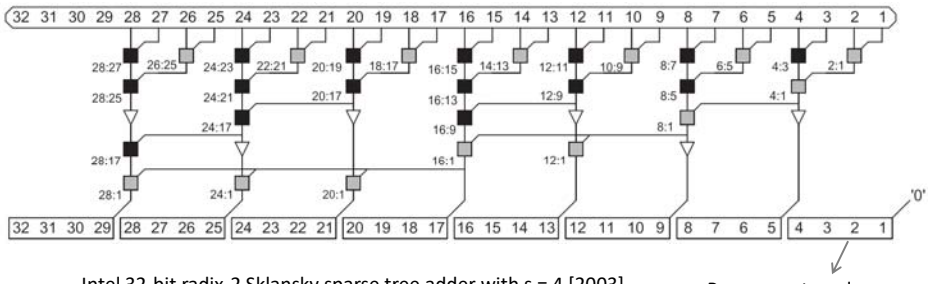
(26:24) + (23:21) + (20:18) + (17:15) + (14:12) + (11:9) + (8:6) + (5:3) + (2:0)

$N = \log_3(n)$

30

Sparse Tree Adders

- Combine PG tree and carry-select
  - Make a sparse PG Tree to only compute carries into short groups (s = 2, 4, 8, or 16 bits)
  - Use pairs of s-bit adders to precompute the sums
  - Use mux to select the correct sum based on carries from the PG tree

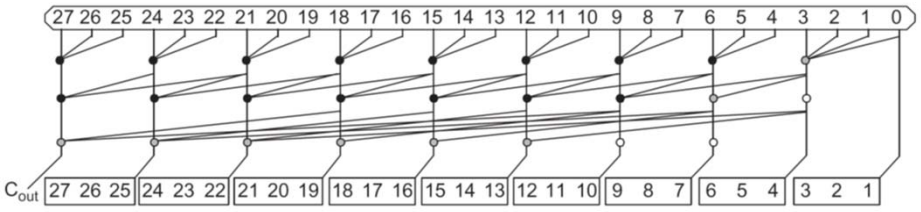


Intel 32-bit radix-2 Sklansky sparse tree adder with s = 4 [2003]

Pre-compute only sum

Sparse Tree Adders

- Small number of logic levels, lower gate count and power consumption
- Widely used in high-performance 32-64-bit high radix adders



Radix-3 Kogge-Stone sparse tree adder with s = 3

- Sparse
- High radix
- Carry save adder